**Advanced Computer Communication (COMP3002)**

**CURTIN UNIVERSITY**
**School of Electrical Engineering, Computing and Mathematical Sciences**
**Discipline of Computing**

**Assignment 2**

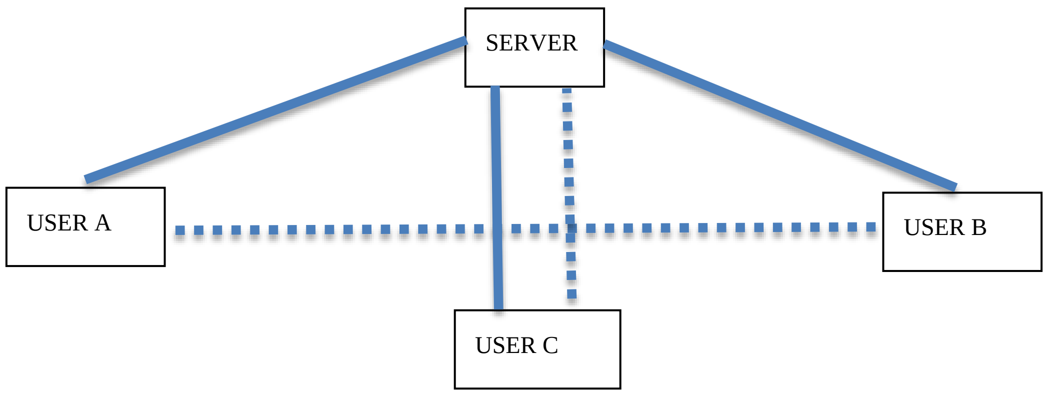
# My Network Chat System

**Due Date:** **4pm, Monday, 2 November, 2020**

## Objective:

The objective of this assignment is to develop two complementary programs (a **USER** and a **SERVER**) that implement an application, called My Network Chat System (MNCS). Note that MNCS **does not** follow the Internet Relay Chat (IRC) specified in RFC 1459 and its revisions in RFC 2810, RFC 2811, RFC 2812 and RFC 2813. For a brief explanation about IRC, you can read The IRC Prelude: http://www.irchelp.org/irchelp/new2irc.html.

## Overview



The figure shows the MNCS server (SERVER) and three MNCS users (USER). A user (e.g., you) uses a program (USER) to connect to a server (SERVER).

In IRC, the SERVER contains an interconnected servers and each user is connected to one of the servers. Further, a user can join to one or more channels (chatting topics) to communicate with one or more other users joining the same channels. Once it joins a channel, a user, e.g., USER A, can send a message to one or more users in the same channel, e.g., USER B and USER C (shown by the dashed lines), through the server (shown by the solid lines) in the figure. As an example, USER A can send a message "Hello" to USER B by sending the message to the SERVER which will send that "Hello" to USER B.

In MNCS, the SERVER has only one server that contains **a set of pre-threaded threads**, each of which serves one connected USER. MNCS has **only one channel**. Like in IRC, a

USER in MNCS (e.g., USER A) must connect to the SERVER to join the system. Once it is connected, the USER can send commands to the SERVER. The SERVER recognizes only the following MNCS commands. Note that **MNCS commands do not exactly the same as the IRC commands.**

**<u>MNCS Commands</u>**

a. **JOIN <username> <realname>** is used at the beginning of a connection to specify the username and realname of a new user. For example,

   **JOIN John Elway John**

   A username has a maximum length of 10 (ten) characters. A real name has a maximum of 20 characters that may include spaces.

   Receiving this command, the server echoes this command to the user, in addition to the host name of the user. Thus, the user's screen may show the following text:

   **JOIN John Elway John**
   **SERVER: JOIN John – John Elway - lab218-a01.cs.curtin.edu.au**

b. **NICK <nickname>** is used to give a user a unique nickname or to change its existing nickname. Each nickname has a maximum length of 9 (nine) characters.

   Receiving the command, the server checks if it is a duplicate nickname; that is, another user has the same nickname. If it is, the server sends an error message to the user. Otherwise, the server sends an acknowledgment message to the user. The user prints the message from the server on its screen. Note that, the nickhame of a user is initialized with his/her username.

   As an example, a screen of a user may show the following text:

   NICK Ben
   SERVER: Your new nickname is Ben

   **or**

   NICK Bob
   SERVER: You have changed your nickname from Ben to Bob

c. **WHO** is used to ask information about users currently in MNCS.

   The server sends to the user detailed information of all users currently in the system, i.e., their nicknames, usernames, real names, hostnames, and the time durations (in seconds) the users have been in the system.

   For example, a user's screen may show the following text:

```
WHO
SERVER: Red John John Elway Lab218-a01.cs.curtin.edu.au  100
SERVER: Ben Alan Alan Shaw Lab218-b01.cs.curtin.edu.au 2000
```

d. **WHOSIS <target>** is used to ask the real name and host name of a user with nickname **<target>**. The server responds with an error if there is no user with nickname **<target>** in the system. Otherwise, the server sends the real name and host name of the user whose nickname is **<target>**. The user prints the respond on the screen.

e. **TIME** is used to query the local time in the server. The server sends to the user its local time. The user shows the time on the screen.

f. **PRIVMSG <target> <text>** is used to send text messages between users. **<target>** can be either the nickname or the user name of the recipient of the message. Each **<text>** has a maximum length of 256 characters terminated by a carriage return – line feed pair.

   A user sends this command to the server. Receiving the command, the server sends the **<text>** together with the sending user's nickname to a user with nickname **<target>**.

   The user that receives the nickname and the **<text>** prints them to the screen in a new line. As an example, the screen of a user with nickname Ben that receives a message from user Star may show the following text:

```
PRIVMSG Star: Hello Ben. Hope you like Assignment2!
```

g. **BCASTMSG <text>** is used to send a broadcast message to other users. Each **<text>** has a maximum length of 256 characters terminated by a carriage return – line feed pair.

   A user sends this command to the server. Receiving the command, the server sends the nickname and the **<text>** to each other user in MNCS.

   Each user that receives the nickname and the **<text>** prints them to the screen in a new line. As an example, the screen of a user with nickname Ben that receives a BCASTMSG from user Star may show the following text:

```
BCASTMSG Star: Hi all! Hope you like this MNCS assignment!
```

h. **QUIT <text>** The user is terminating his/her session.

   The server (i) sends a "reply message" and closes the connection to the user, and (ii) sends the user's nickname and **<text>** to the remaining users, stating that the user has closed connection. If there is no **<text>**, the server uses a "default message".

   For example, the screen of user Bob that uses the QUIT command may show the following text:

```
QUIT Bye all!
SERVER:  You have been chatting for 100 seconds. Bye Bob!
```

All other user’s screens may show the following text:

SERVER: Bob is no longer in our chatting session.
Bob’s last message: QUIT Bye all!

**or**

SERVER: Bob is no longer in our chatting session.
There is no last message from Bob!

**Error handling**

a. The SERVER is responsible for checking the validity and syntax of its USER’s command. Similarly, the MNCS USER should also check the validity/syntax of each user command.

b. The SERVER terminates a user if the server does not receive any command from the user within wait_time seconds. Before terminating a user, the server sends a message to the terminated user and to other remaining users. The valid number for wait_time is an integer between 1 and 120 seconds.

c. Make sure to check for error return from each system call, and to close every socket created.

d. Proper handling of mutual exclusion and race conditions is required.

## **Implementation:**

1. Your program must be written in 'C' using TCP/IPv4 sockets. The program must run on the Discipline of Computing's laboratory computers.

2. You are allowed to use any existing functions/codes. However, you are responsible for the correctness of the functions, and you have to mention/cite the sources.

3. The MNCS server is a pre-threaded server; initially the server creates *n* threads. For each user, the server uses one available thread from a pool of threads that you have created when you started your server. The server creates *m* additional threads if there is only one thread available in the pool.

4. The server is run as

   ./server *n* *m* wait_time

5. The MNCS user should be started with either:

   user Server_IP_address port

   **or**

   user Server_host_name port

   Note, IP_address is the MNCS server's address (in dotted decimal notation).

   Once the MNCS user is connected to the MNCS server, the MNCS user waits for user command, and a prompt "**USER**>" should be shown on the screen.

6. You are allowed to make your own assumptions/requirements other than those already given. However, YOU HAVE TO DOCUMENT ANY ADDITIONAL ASSUMPTIONS/LIMITATIONS FOR YOUR IMPLEMENTATIONS.

## <u>Instruction for submission:</u>

1. Assignment2 submission is **compulsory**. Students will be penalized by a deduction of ten percents per calendar day for a late submission. **An assessment more than seven calendar days overdue will not be marked and will receive a mark of 0**.

   **Due dates and other arrangements may only be altered with the consent of the majority of the students enrolled in the unit and with the consent of the lecturer.**

2. You must (i) put your program files, i.e., ., **server.c**, **user.c**, makefiles, and other required files, in your home directory named ACC/assignment2, (ii) submit the soft copy of assignment **report** to the unit Blackboard (**in one zip file)**, i.e., **ID_Assignment2.zip**.

3. The soft copy of your **report**, as stated in 2(ii), MUST include:

   - A signed cover page that explicitly states the submitted assignment is **your own work**. The cover includes the words "Advanced Computer Communication Assignment 2", and your name in the form: family, other names. Your name should be as recorded in the student database.

   - Software solution of your assignment that MUST include (i) **all source code** for the programs with proper in-line and header documentation. Use proper indentation so that your code can be easily read. Make sure that you use meaningful variable names, and delete all unnecessary comments that you created while debugging your program; and (ii) **readme file** that, among others, explains how to compile your program and how to run the program.

   - Detailed discussion on all shared data structures used in the server, and how any mutual exclusion / thread synchronization is achieved on shared resources, e.g., memory, and what threads access the shared resources.

   - Description of any cases for which your program is not working correctly or how you test your program that make you believe it works perfectly.

   - Sample inputs and outputs from running your programs.

   **Your report will be assessed (worth 20% of the overall mark for Assignment 2).**

4. Demo requirements:

   - You will be required to demonstrate and explained your programs.
   - You MUST keep the source code for your programs in your home directory, and the source code MUST be that submitted.
   - The programs MUST run on our Department's computer system.

**Failure to meet these requirements may result in the assignment not being marked.**